REMARKS

Claims 1-4 and 6-17 are pending in the application.

Claim Rejections

In the present Office Action, claims 1-4, 6-11, and 13-17 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharangpani (US patent No. 5,669,537, hereinafter "Sharangpani") in view of Sollars (US patent No. 5,900,025, hereinafter "Sollars"). Applicant has carefully considered the examiner's new rejections and the cited art. However, Applicant believes the claims recite features neither disclosed nor suggested. Accordingly, Applicant traverses the above claim rejections and requests reconsideration in view of the following discussion.

Claim 1 is repeated here for reference purposes.

"A pipelined multistreaming processor, comprising:
an instruction cache for concurrently providing a plurality of instructions for a plurality of instruction streams;
fetch logic coupled to said instruction cache enabled to concurrently fetch said plurality of instructions for said plurality of instruction streams from said instruction cache;
a plurality of instruction queues coupled to said fetch logic where each one of said plurality of instruction queues is associated with at least one of said plurality of instruction streams, wherein the number of said plurality of instruction queues is greater than said plurality of instruction streams that are provided by said instruction cache;
a dispatch stage coupled to said plurality of instruction queues for selecting and dispatching instructions for said plurality of instruction streams to a set of execution units; and
select logic coupled to said instruction cache, and to said plurality of instruction queues, said select logic monitoring each of the plurality of instruction queues, said select logic selecting ones of said plurality of instruction streams to fetch instructions from said instruction cache, the selecting based on the monitoring."

Regarding claim 1, in paragraph 8 of the present Office Action, it is suggested state that:

> "Sollars taught plural instruction queues that are selected to receive instructions from cache and the output instructions are selected to be sent to execution unit (e.g., see fig. 15 of Sollars). Sollars taught that some streams were active and some inactive (as discussed above). Sollars taught that operation of monitoring other queues in the system namely the request queues to determine if it was empty (e.g., see col.12, lines 42-58). Consequently, one of ordinary skill would have been motivated to monitor queues used in the Sollars system for determining actions to be taken regarding the queued data or instructions."

However, Applicant respectfully submits that the monitoring referred to in Sollars is quite different from that recited and does not disclose or suggest the claimed "monitoring each of the plurality of instruction queues, said select logic selecting ones of said plurality of instruction streams to fetch instructions from said instruction cache, the selecting based on the monitoring." For example, Sollars discloses

> "As shown in FIG. 11b, upon detecting each unforced deallocation of a context level control register set 104 (i.e., normal termination of a context), step 218, the operating system determines if the context request queue is empty, step 220. If the queue is empty, the operating system takes no further action. On the other hand, if the queue is not empty, the operating system dequeues a pending context request and allocates the free context level control register set 104 to the dequeued context, step 222." (Sollars, col. 12, lines 42-50).

As can be seen from the above, Sollars's operating system determines if the context request queue is empty, and if so, takes no further action. If the queue is not empty, the operating system dequeues a pending context request. It is noted that a context request queue is quite different from an instruction queue. Sollars describes a single context request queue whereas multiple parallel instruction queues are described. Therefore, monitoring of a context request queue is a simple matter of determining whether or not there are any requests to be dequeued, whereas monitoring of multiple parallel instruction queues would involve more complex prioritization decisions. Sollars teaches no such features. In addition, it is noted that claim 1 recites "monitoring each of the plurality of

instruction queues, said select logic **selecting ones of said plurality of instruction streams to fetch instructions from said instruction cache, the selecting based on the monitoring**." In contrast, Sollars teaches dequeueing based on monitoring, rather than fetching instructions for selected streams based on monitoring.

It is further suggested on page 5, paragraph 8 of the present Office Action that

> "Further since when a queue was not empty more instructions would have been waiting to be executed and when the queue was full no more instructions could have been fetched to the queue without losing and instruction then monitoring the queues would have been obvious to one of ordinary skill at least to ensure that all instructions queued were executed and there was room in a queue when an instruction was fetched to the queue. Also since the Sollars queue stored instructions for separate streams then clearly when one queue for a stream was full then one of ordinary skill would have been motivated to select another stream. This operation provides for the fetching of threads based on monitoring."

However, claim 1 recites, in relevant part "selecting ones of said plurality of instruction streams to fetch instructions from said instruction cache, the selecting based on the monitoring." As may be appreciated, it is the selecting of ones of the instruction streams that is based on monitoring. Applicant submits that selecting instruction streams is different from simply avoiding data loss. For example, Sollars does not suggest any mechanism for choosing among plural queues that are not empty for fetching instructions from plural streams. Nor would such a mechanism be obvious given the disclosure of Sollars. Accordingly, Applicant finds no teaching or suggestion is Sollars, nor in the combination of Sollars and Sharangpani, of "monitoring each of the plurality of instruction queues, said select logic selecting ones of said plurality of instruction streams to fetch instructions from said instruction cache, the selecting based on the monitoring," as is recited in claim 1.

For at least these reasons, Applicant submits that claim 1 is patentably distinguished from the cited art. In addition, as independent claim 7 includes similar features, claim 7 is believed patentably distinguished for similar reasons.

Also, regarding claim 1, on page 3, paragraphs 4 and 5 of the present Office Action, it is suggested that

> "Sharangpani did not expressly detail (claims 1, 7) that the instructions that were fetched concurrently were for a plurality of threads and the queue were for a plurality of threads. Sollars however taught concurrently fetching instructions from plural threads from cache (16) to a plurality of queues (254a-254d)(e.g., see fig. 15) and selecting instructions from the queue using selector (256) and sending the instructions to the execution units (e.g., see fig. 15)(e.g., see col.2, line 15-col. 3, line 10 and col. 6, lines 16-35 and col. 14, lines 1-40).
>
> It would have been obvious to one of ordinary skill in the DP art to combine the teachings of Sharangpani and Sollars. Both references were directed toward the problems of fetching and providing instructions to plural execution units in parallel. One of ordinary skill would have been motivated to incorporate the Sollars teachings of fetching instructions for plural threads to plurality of queues and then selecting queue for dispatch to execution units at least to reduce the time needed to fetch instructions to execution units and effectively decoupling the fetch of instructions from the dispatch of instructions to execution units (e.g., see col. 14, lines 25-36)."

To establish a *prima facie* case of obviousness, the proposed modification cannot change the principle of operation of a reference, nor render the prior art unsatisfactory for its intended purpose. MPEP 2143.01. It is first noted that the examiner has not made clear what teachings of Sollars are to be included in the combination of Sharangpani and Sollars in order to arrive at the claimed subject matter. Nevertheless, assuming all of the elements necessary to implement Sollars teachings of "fetching instructions for plural threads to plurality of queues and then selecting queues for dispatch to execution units," as suggested in paragraph 5 of the present Office Action, one would have to replace at least the instruction execution front-end 301 and the in-order dispatch queues 222 of figure 4 of Sharagpani with the IFU 12 of Sollars. However, Sharangpani and Sollars have completely different principles of operation. Sharangpani discloses:

> "The processor 201 includes chain-building and steering logic 220, as well as in-order dispatch queues 222, to provide for efficient dynamic

> instruction scheduling and execution. The details of the chain-building and steering logic 220 and the in-order dispatch queues 222 are provided below in reference to FIG. 4." (Sharangpani. col. 4, lines 57-62).
>
> "Chain-building and steering logic coupled to the dispatch queues identifies a consumer instruction relying on a producer instruction for an operand, and issues the consumer instruction to the same dispatch queue as the producer instruction that it is dependent upon." (Sharangpani, Abstract).

As may be seen from the above, Sharangpani is directed to efficient scheduling and execution of chains of dependent instructions. In contrast, Sollars discloses:

> "IFU 12 further comprises selector 256 for selectively dispatching decoded instructions queued in queues 254a-254d to execution units 14, and control circuitry 258 for controlling buffers 250a-250d, 252a-252d, 254a-254d, and selector 256. Control circuitry 258 allocates these resources to the various threads and MLRs in accordance with the execution priority levels of the threads/MLRs and their contexts, stored in their respective context and thread level control register sets 104 and 106.
>
> . . . the "extra" context/thread level control register sets 104/106 and macro-trap level partitioned subsets 107 allow the state of the "suspended" threads/MLRs to be kept on chip, thereby facilitating much faster execution resumption as control circuitry 258 is able to "reallocate" the fetch/decode/queue paths to some of the "suspended" threads/MLRs." (Sollars, col. 14, lines 5-13 and 30-36).

As may be seen from the above, Sollars is directed to allocating resources to various threads and MLRs in accordance with the execution priority levels of the threads/MLRs and their contexts, stored in their respective context and thread level control register sets, without regard to instruction dependencies. The suggested combination of references would require a substantial reconstruction and redesign of the elements shown in Sharangpani, removing Sharangpani's chain building and steering logic, and replacing it with Sollars's context and thread level control register sets and entirely changing the principle of operation of Sharangpani. For at least these reasons, Applicant submits not only would the proposed combination not result in Applicant's presently claimed invention, but there is no suggestion or motivation to make the

suggested modification(s), and a prima facie case of obviousness has not been established.

Further, Applicant submits that the suggested modifications would render Sharangpani unsuitable for its intended purpose. Removal of the instruction execution front-end 301 from Sharangpani, and the included chain building and steering logic 220, removes the elements responsible for ensuring that chains of dependent instructions are grouped together and steered to the appropriate execution units. The proposed combination would not have the elements necessary to group chains of instructions together and steer them to the same execution units, as intended by Sharangpani. For at least these additional reasons, a prima facie case of obviousness has not been established.

Still further, Applicant submits that the suggested motivation to "incorporate the Sollars teachings of fetching instructions for plural threads to plurality of queues and then selecting queue for dispatch to execution units . . .**effectively decoupling the fetch of instructions from the dispatch of instructions to execution units**", as quoted above from paragraph 5 of the present Office Action, is not found in the references. Rather, it appears to be a product of hindsight in view of Applicant's disclosure. There is nothing in the cited references which suggests the desirability of such a decoupling.

Accordingly, Applicant submits that a *prima facie* case of obviousness is not established and claim 1 is believed patentable over the cited art for at least these additional reasons.

In addition to the above, claim 12 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over Sharangpani and Sollars as applied to claims 1-4, 6-11, and 13-17 above, and further in view of Hirata (U.S. Patent 5,430,851). As claim 12 is patentable for at least the reasons given above, further discussion of the features of this claim is believed unnecessary at this time.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

Respectfully submitted,

/James W. Huffman/

James W. Huffman
Reg. No. 35,549
ATTORNEY FOR APPLICANT(S)

Date: _____6/13/06_____